A few things I would like to know (empirically)

Bertrand Meyer

ESEM 2010, Bozen

Summary

Goals for empirical research,
as seen by a software engineer

Empirical work from ETH

With Karine Arnout: spotting hidden contracts in .NET libraries, IEEE Computer, Nov. 2003

- Are contracts a figment of Eiffel programmers' imagination?

With Ilinca Ciupa, Andreas Leitner and Manuel Oriol: assessment of random testing, ISSTA 2007

- How good is random testing at finding faults?

With Ilinca Ciupa, Lisa Liu, Manuel Oriol, Andreas Leitner and Raluca Borca-Muresan: evaluation of test failure results, RAF 2007

- Human assessment of failure kinds

Details: se.ethz.ch/~meyer/publications

Empirical work from ETH

With Lisa Liu and Bernd Schoeller: boolean queries, TAP 2007

- A novel way of partitioning the object state, based on boolean queries

With Andreas Leitner, Manuel Oriol, Ilinca Ciupa and Andreas Zeller: test case minimization, ASE 2007

- Producing simpler test cases

With Marie-Hélène Nienaltowski and Michela Pedroni: Compiler error messages for novices, SIGCSE 2008

- How can the design of compiler error messages help novices?

Empirical work from ETH

With Ilinca Ciupa, Alexander Pretschner, Andreas Leitner and Manuel Oriol: predictability of random tests, ICST 2008

- How sensitive are random tests to the parameters chosen?

With Nadia Polikarpova and Ilinca Ciupa: contract inference for contract-equipped languages, ISSTA 2009

- Eiffel interface for Daikon: what contracts can tool infer, what contracts do programmers write, and how do they compare?

Empirical work from ETH

With 5 coauthors: Programs that Test Themselves, IEEE Computer, Sept. 2009

- Contract-based push-button testing (AutoTest), including test generation, test oracles and test extraction from failures

With Christine Choppy (Paris-XIII), Jørgen Staunstrup (IT U. of Denmark), Jan van Leeuwen (Utrecht): Research Evaluation for Computer Science, Comm. ACM, April 2009

- How to evaluate CS researchers (hint: not the Web of Science)

With Yi Wei, Yu Pei, Carlo Furia, Stefan Buchholz and Andreas Zeller (Sarrebruck): automatic fixing of programs with contracts, ISSTA 2010

- Automatically generated fixes correspond to those produced by programmers in about a third of cases

Empirical work from ETH

With Martin Nordio and Roman Mitin: distributed software engineering course, ICSE 2010 (also: with Carlo Ghezzi, Elisabetta di Nito and Giordano Tamburelli (Politecnico di Milano), SEAFOOD 2009)

- Techniques, in particular strict enforcement of contracts, that help distributed projects succeed

With Yi Wei, S. Gebhardt, Manuel Oriol: Precondition satisfaction, ICST 2010

- Techniques for making automatic testing complete

Empirical work from ETH

With Michela Pedroni and Manuel Oriol: What beginning CS majors know, se.ethz.ch/~meyer/publications/teaching/background.pdf

- Initial knowledge of entering CS students, collected over seven years

With Sebastian Nanz and Michela Pedroni: comparison of pedagogical approaches to concurrency

- With what model do students learn better and make fewer mistakes?

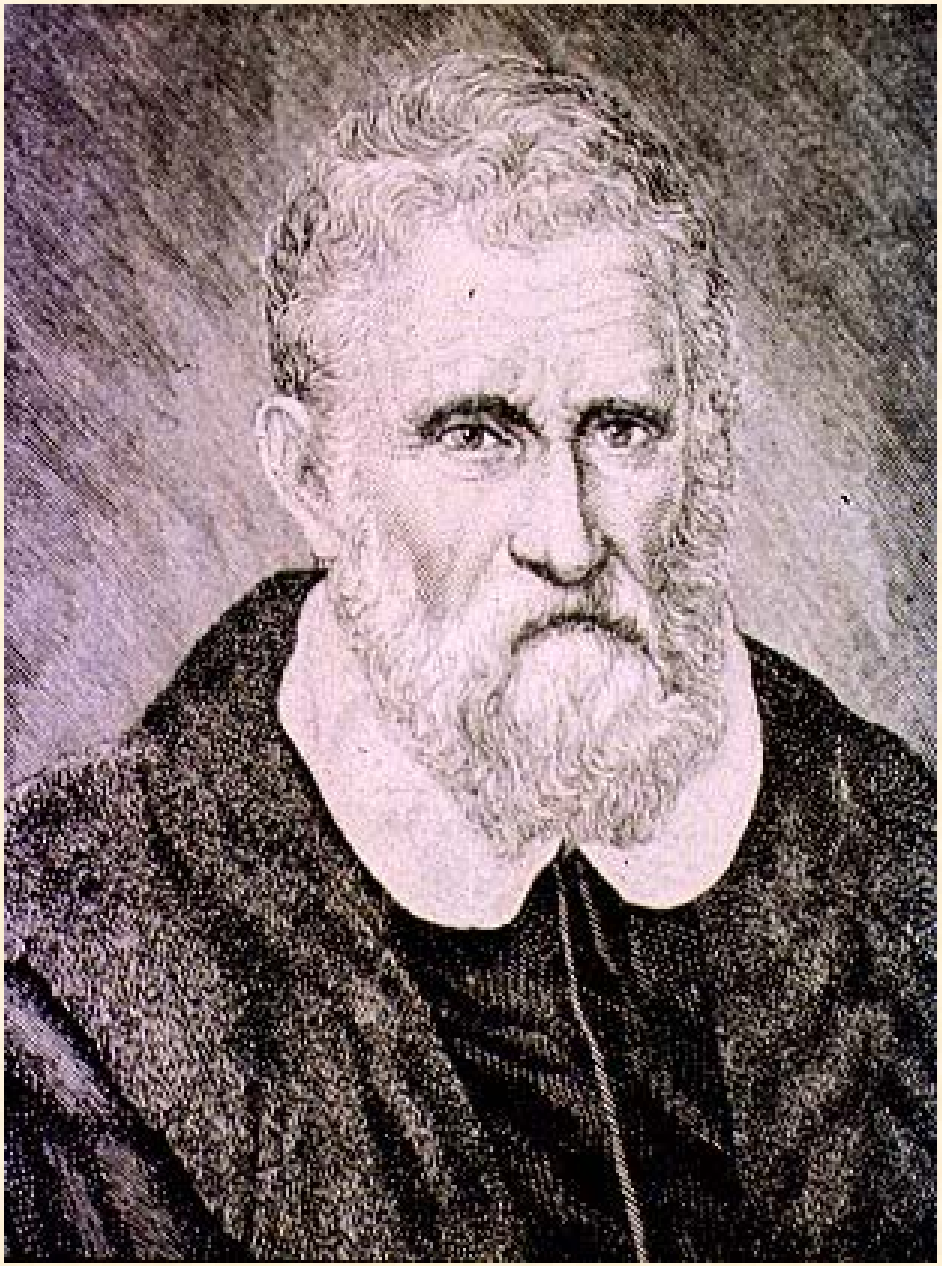
Details: se.ethz.ch/~meyer/publications

"Great" ideas

Structured programming
Object-oriented programming
Design by Contract
Object-oriented analysis
Seamless development
Test-driven development
Model-driven architecture
UML
Use cases
Pair programming
Refactoring
Scrum
Aspect-oriented programming

How do we know they work?

Limits of deductive approaches



(after R. Lister, after D. Valentine)

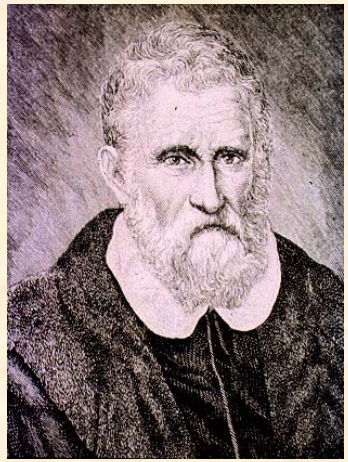
The Marco Polo principle:

"I traveled far and saw wonderful things"

Marco Polo papers

Raymond Lister: *After the Gold Rush: Toward Sustainable Scholarship in Computing*, 10th Austr. comp. conf., 2008 (abridged)

Valentine explained that authors of *"Marco Polo papers"* describe how they tried a new curriculum, adopted a new language, put up a new course. They describe the reasoning, explain components, and draw a conclusion such as *"Overall, I believe the approach has been a big success"* or *"Students seemed really to enjoy it "*.



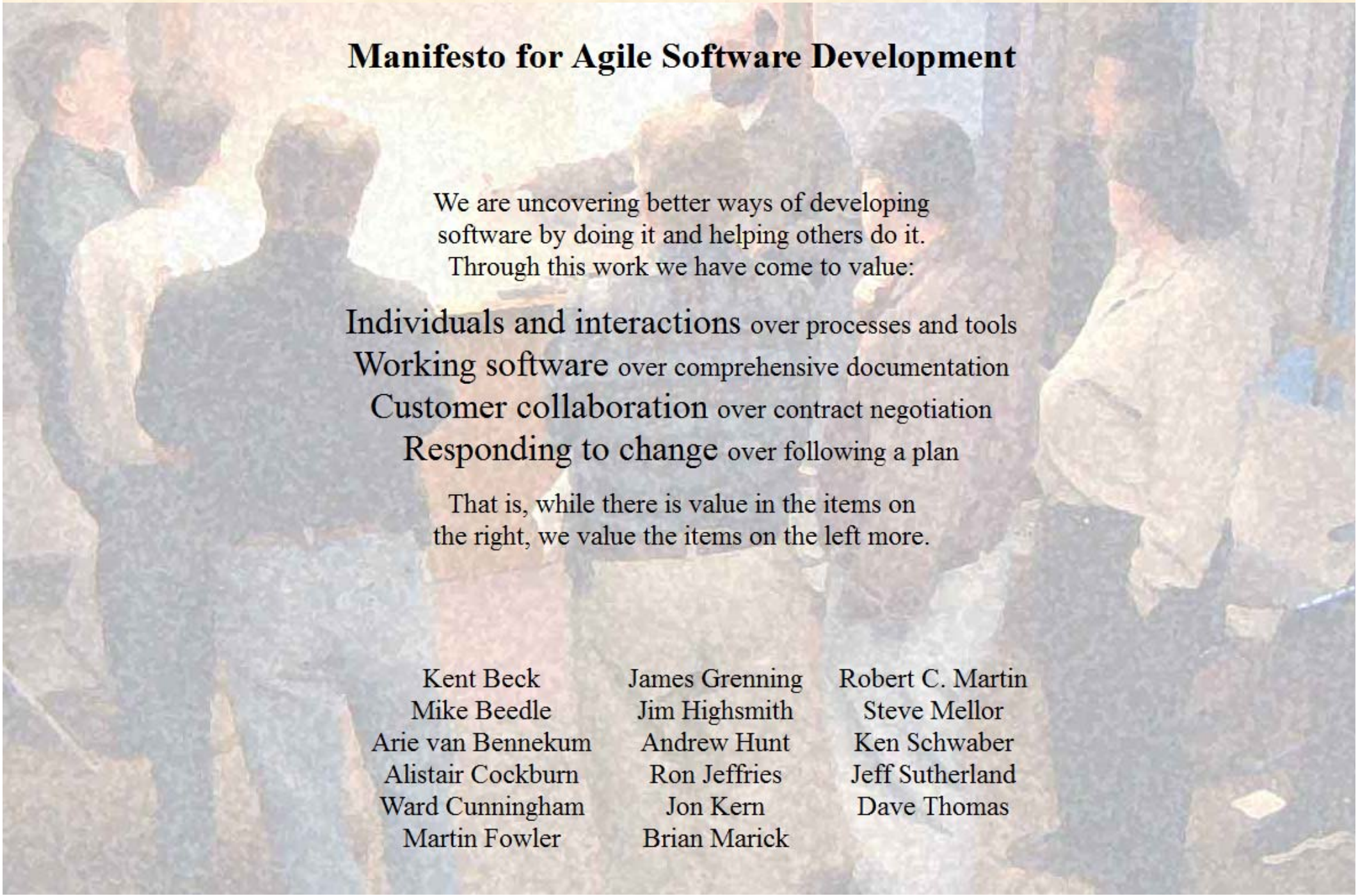
He went on:

It seems that with just a little more effort at providing evidence we could wring a great deal more benefit from the exercise.

Example statement (Dijkstra, 1968)

"For a number of years I have been familiar with the observation that the quality of programmers is a decreasing function of the density of ***go to*** *statements in the programs they produce. More recently I discovered why the use of the* ***go to*** *statement has such disastrous effects, and I became convinced that the* ***go to*** *statement should be abolished from all "higher level" programming languages... At that time I did not attach too much importance to this discovery; I now submit my considerations for publication because in very recent discussions in which the subject turned up, I have been urged to do so."*

The Agile manifesto



Manifesto for Agile Software Development

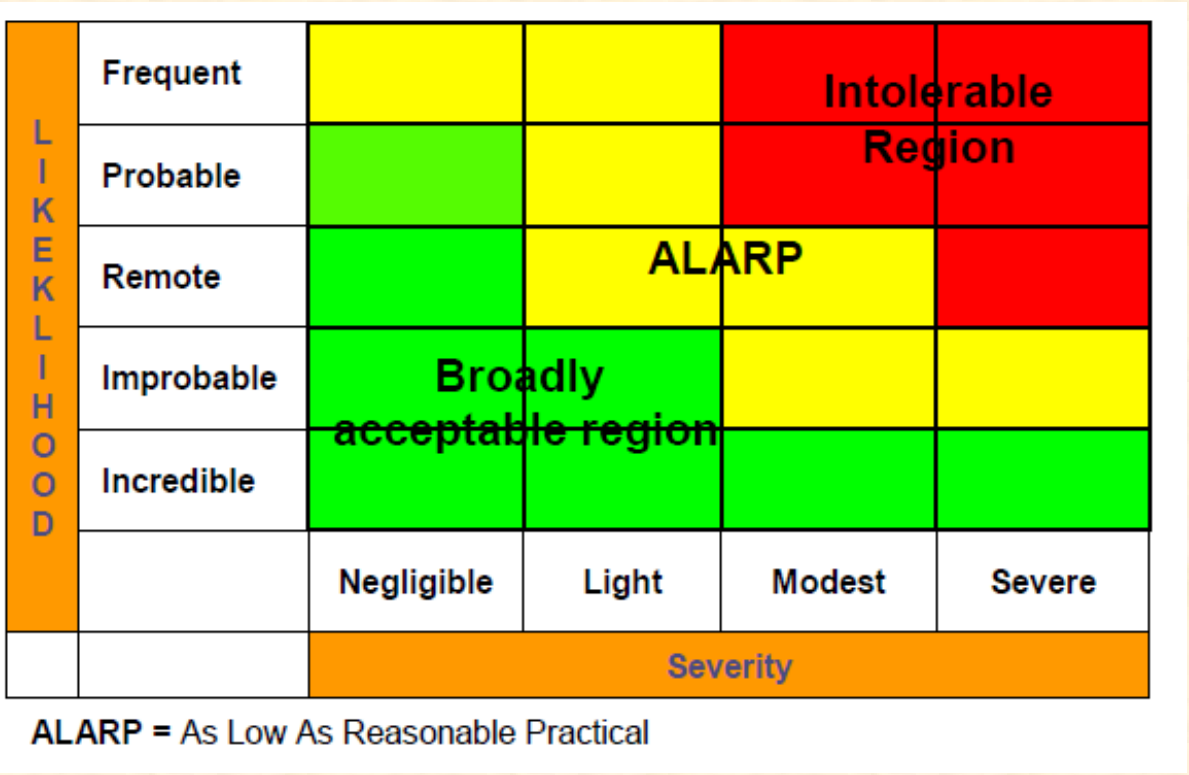
We are uncovering better ways of developing
software by doing it and helping others do it.
Through this work we have come to value:

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

That is, while there is value in the items on
the right, we value the items on the left more.

Kent Beck
Mike Beedle
Arie van Bennekum
Alistair Cockburn
Ward Cunningham
Martin Fowler
James Grenning
Jim Highsmith
Andrew Hunt
Ron Jeffries
Jon Kern
Brian Marick
Robert C. Martin
Steve Mellor
Ken Schwaber
Jeff Sutherland
Dave Thomas

How the rest of the world views software



Software (IEC 62304):

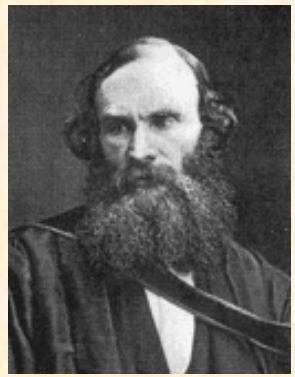
LIKELIHOOD = 100%

ISO 14971 (medical devices):

Risk = f (*LIKELIHOOD*, *Severity*)

Source: C. Gerber, Stryker Navigation

Measurement



"To measure is to know"

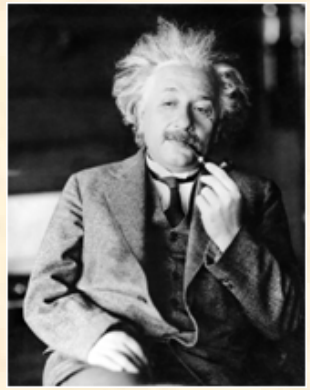
"When you can measure what you are speaking about and express it in numbers, you know something about it; but when you cannot measure it, when you cannot express it in numbers, your knowledge is of a meager and unsatisfactory kind; it may be the beginning of knowledge, but you have scarcely in your thoughts advanced to the state of *Science*, whatever the matter may be. "

"If you cannot measure it, you cannot improve it."

Lord Kelvin

"You can't control what you can't measure"

Tom de Marco



"Not everything that counts can be counted, and not everything that can be counted counts."

Albert Einstein (attributed)

What the field needs

Two complementary views:

- Deductive:
  "Try my approach!"

- Inductive:
  "I tried this and it
  - *Worked!*
  - *Didn't work!"*

Compare with research in physics:

- Theoretical
- Experimental

A horror story

Semicolon as:

- Separator (Algol):

  p ; q ; r

  -- As in: f (x, y, z)

- Terminator (C):

  p ; q; r;

Why do Ada, C++, Java, C#...
use terminator convention?

Answer: Gannon & Horning, *Language Design for Programming Reliability*, IEEE Trans. on S.E., June 1975

Experiment: programmers in language with terminator convention make fewer mistakes

The mistakes that do happen in practice

```
while (e) ;
        a
```

```
if (e) then
        a
else ;
        b
```

A horror story

Semicolon as:

- Separator (Algol):

  p ; q ; r

- Terminator (C):

  p ; q; r;

-- As in: f (x, y, z)

Why do Ada, C++, Java, C#... use terminator convention?

Wrong!

- **Syntax errors only**
- **PL/I-trained programmers**
- **In separator language, extra semicolon is error!**

Answer: Gannon & Horning, *Language Design for Programming Reliability*, IEEE Trans. on S.E., June 1975

Experiment: programmers in language with terminator convention make fewer mistakes

Empirical software engineering

Advocated for many years by such people as Barry Boehm, Vic Basili, Walter Tichy, Watts Humphrey, Dieter Rombach…

Aim: subject software engineering claims to rigorous experimental evaluation

Many more papers recently: ICSE, ESEC, ESEM

Early empirical papers

Industry: not reproducible

University: not credible

What has changed

1. In the past ten years, the availability of large open-source project repositories has provided empirical software engineering researchers with a wealth of objective material that makes verifiable, repeatable analyses possible

2. Some commercial software has also become available for examination, e.g. from Microsoft

3. Standards have risen, and there is now a small base of credible empirical papers

Empirical SE papers, today

Better than they used to be, but:

- Often very disappointing, e.g. many studies ask people what they think instead of using objective measures
- "**Threats to Validity**" section kills generalization

"Threats to validity"

"*While we believe that the comments from the 3 examined OSs well represent comments in systems software, we do not intend to draw any general conclusions about comments in all software. Similar to any characteristic study, our findings should be considered together with our evaluation methodology. Section 7 presents a preliminary study of comments in other large software written in C++ and Java. Since we examined comments manually, subjectivity is inevitable. However, we tried our best to minimize such subjectivity by using double verification.*"

(Padioleau, Tan, Zhou, *Listening to programmers - Taxonomies and characteristics of comments in operating system code*, ICSE 2009)

In other disciplines

Example from a medical textbook*:

A woman (24 years of age; height: 1.70 m; weight: 60 kg) is in hospital due to tremendous thirst. She drinks large amounts of water and produces over 10 liters of urine each day. Suspected diagnosis is diabetes insipidus. The vasopressin concentration in plasma (measured by RIA method) is 10 fmol per liter.

1. Calculate the secretion of vasopressin (mg/hour) from the neurohypophysis of a normal 60-kg person and of this patient.
4. Estimate the relation between this concentration and that of a healthy individual.
5. Does this ratio have implications for the interpretation of her special type of diabetes insipidus?
6. Is it dangerous to lose 10 litres of urine per day?

*www.mfi.ku.dk/ppaulev/chapter26/Chapter%2026.htm,*abridged*

Proposal 1: openness

Better refereeing process

- Experimental work acceptable
- Reproducibility papers acceptable
- "No surprise" dismissal not valid

Openness

- All code and data available on Web
- All assumptions disclosed

Reproducibility

No exaggerated "Threats to Validity" excuses

Proposal 2: SEERM

Software Engineering Experiment Reproduction Market

(model: Odesk.com)

"Buy":

- Offer experiment for reproduction
- Disclose as little or as much as desired
- Offer conditions (i.e. share in publication, or not)
- Offer specified number of points (which you must have!)

"Sell":

- Offer to reproduce experiments
- Specify number of points
- Disclose as little or as much as desired
- Offer conditions (i.e. share in publication, or not)

Proposal 3: obligatory post-mortem

Example: Ariane

Should be required!

Example: black boxes in aviation

Proposal 4: call for answers

Select 10 questions of general interest

Derive generally agreeable, empirically validated answers

Revalidate regularly

My hunch: start with process questions

10 answers to 10 questions

Select ten questions

Assemble panel of experts

Publicize questions, invite answers

Publication date: September 2010

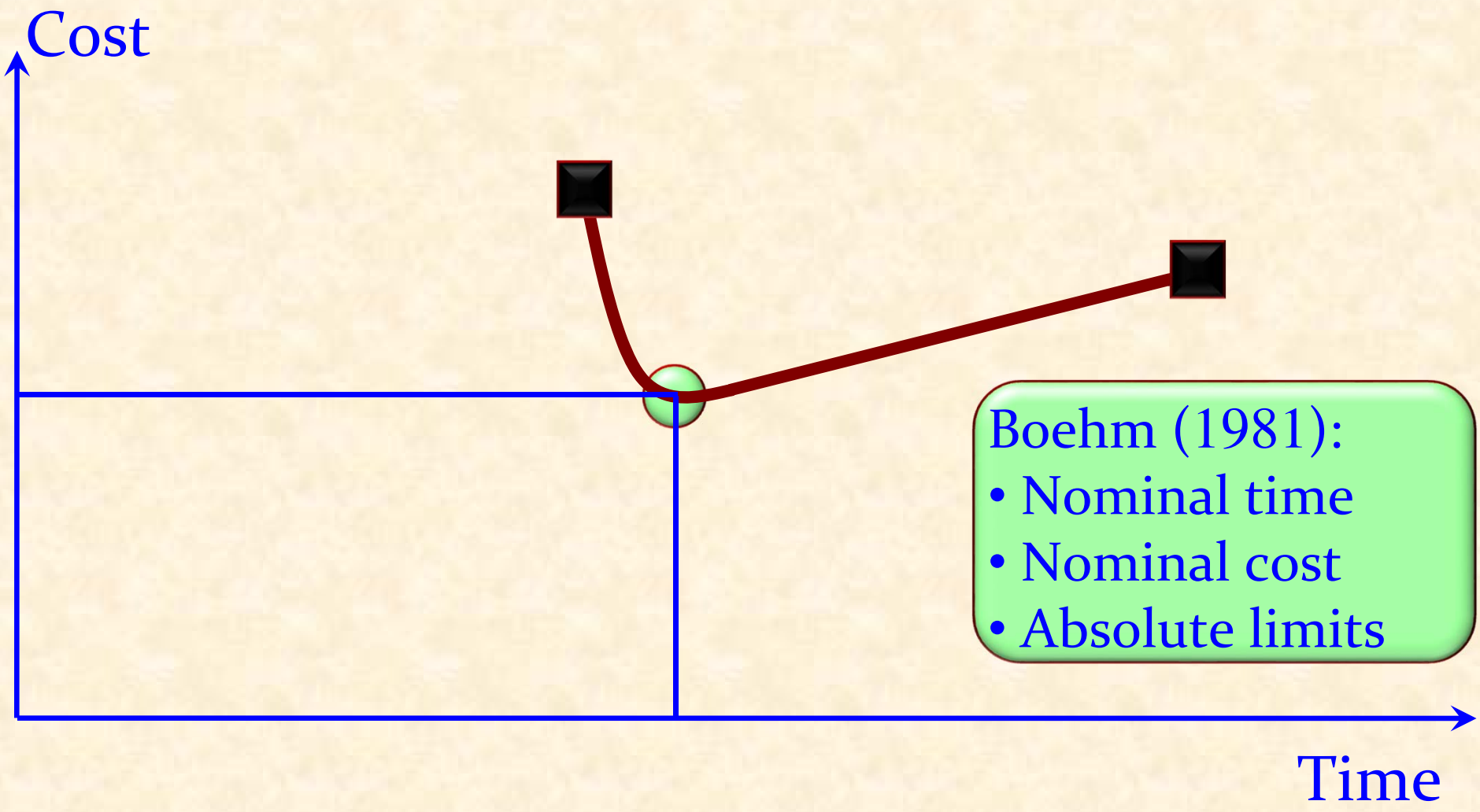
Submission date: February 2011

Session: TOOLS EUROPE, June 27-July 1, 2011 (Zurich)

Sample questions: pair programming

1. Does it lead to fewer bugs?
2. Does it lead to shorter debugging times?
3. Are there good programmers who will not adapt to it?
4. Should it be applied throughout the programming phase?
5. Should it be applied to other tasks, e.g. pair specifying, pair testing?
6. Are there useful variants, e.g. programmer-tester pairing?

Sample questions: nominal values



Sample questions: refactoring

What is better:

- Design?
- Refactoring?
- Some combination?

Sample questions: tests vs specs

What works better:

- Extensive specifications?
- A test-driven process?
- Some combination?

Sample question: RTC vs CTR

Commit strategies:

- Review Then Commit (Google, original Apache)
- Commit To Review (Apache)

See Rigby, German, Storey, *Open Source Software Peer Review Practices: A Case Study of the Apache Server*, ICSE 2008, but need studies on other projects and correlation with software quality measures!

Sample question: complexity measures

Which measures correlate best to quality indicators?

- SLOC
- Function points
- Specific O-O metrics
- McCabe etc.

Sample questions: testing

1. When can we stop testing?

2. Are any coverage measures any good?

3. What is the best distribution between automated and manual test generation?

4. What is the lifespan of a test case?

10 answers to 10 questions

Select ten questions

Assemble panel of experts

Publicize questions, invite answers

Publication date: September 2010

Submission date: February 2011

Session: TOOLS EUROPE, June 27-July 1, 2011 (Zurich)

"Great" ideas

Structured programming
Object-oriented programming
Design by Contract
Object-oriented analysis
Seamless development
Test-driven development
Model-driven architecture
UML
Use cases
Pair programming
Refactoring
Scrum
Aspect-oriented programming

How do we know they work?

References

se.ethz.ch

se.ethz.ch/~meyer

www.bertrandmeyer.com

eiffel.com